C++ STL TUTORIAL

Hope you already understand the concept of C++ Template which we already have discussed in one of the chapters. The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provides general-purpose templatized classes and functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, and stacks.

At the core of the C++ Standard Template Library are following three well-structured components:

Component	Description
Containers	Containers are used to manage collections of objects of a certain kind. There are several different types of containers like deque, list, vector, map etc.
Algorithms	Algorithms act on containers. They provide the means by which you will perform initialization, sorting, searching, and transforming of the contents of containers.
Iterators	Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.

We will discuss about all the three C++ STL components in next chapter while discussing C++ Standard Library. For now, keep in mind that all the three components have a rich set of pre-defined functions which help us in doing complicated tasks in very easy fashion.

Let us take the following program demonstrates the vector container (a C++ Standard Template) which is similar to an array with an exception that it automatically handles its own storage requirements in case it grows:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
   // create a vector to store int
   vector<int> vec;
   int i;

   // display the original size of vec
   cout << "vector size = " << vec.size() << endl;

   // push 5 values into the vector
   for(i = 0; i < 5; i++){
      vec.push_back(i);
   }

   // display extended size of vec
   cout << "extended vector size = " << vec.size() << endl;

   // access 5 values from the vector
   for(i = 0; i < 5; i++){
      cout << "value of vec [" << i << "] = " << vec[i] << endl;
   }

   // use iterator to access the values
   vector<int>::iterator v = vec.begin();
   while( v != vec.end()) {
```

```
      cout << "value of v = " << *v << endl;
      v++;
   }

   return 0;
}
```

When the above code is compiled and executed, it produces following result:

```
vector size = 0
extended vector size = 5
value of vec [0] = 0
value of vec [1] = 1
value of vec [2] = 2
value of vec [3] = 3
value of vec [4] = 4
value of v = 0
value of v = 1
value of v = 2
value of v = 3
value of v = 4
```

Here are following points to be noted related to various functions we used in the above example:

- The push_back( ) member function inserts value at the end of the vector, expanding its size as needed.
- The size( ) function displays the size of the vector.
- The function begin( ) returns an iterator to the start of the vector.
- The function end( ) returns an iterator to the end of the vector.